

Scalability and performance of salinas on the computational plant

Manoj Bhardwaj, Ron Brightwell & Garth Reese
Sandia National Laboratories

Abstract

This paper presents performance results from a general-purpose, finite element structural dynamics code called Salinas, on the Computational Plant (Cplant™), which is a large-scale Linux cluster. We compare these results to a traditional supercomputer, the ASCI/Red machine at Sandia National Labs. We describe the hardware and software environment of the Computational Plant, including its unique ability to support easily switching a large section of compute nodes between multiple different independent cluster heads. We provide an overview of the Salinas application and present results from up to 1000 nodes on both machines. In particular, we discuss one of the challenges related to scaling Salinas beyond several hundred processors on Cplant™ and how this challenge was addressed and overcome. We have been able to demonstrate that the performance and scalability of Salinas is comparable to a proprietary large-scale parallel computing platform.

1 Introduction

Parallel computing platforms composed of commodity personal computers (PCs) interconnected by gigabit network technology are a viable alternative to traditional proprietary supercomputing platforms. Small- and medium-sized clusters are now ubiquitous, and larger-scale procurements, such as those made recently by National Science Foundation for the Distributed Terascale Facility [1] and by Pacific Northwest National Lab [13], are becoming more prevalent. The cost effectiveness of these platforms has allowed for larger numbers of processors to be purchased.

In spite of the continued increase in the number of processors, few real-world application results on large-scale clusters have been published. Traditional large parallel computing platforms have benefited from many years of research, development, and

experience dedicated to improving their scalability and performance. PC clusters have only recently started to receive this kind of attention. In order for clusters of PCs to compete with traditional proprietary large-scale platforms, this type of knowledge and experience may be crucial as larger clusters are procured and constructed.

The goal of the Computational Plant (Cplant™) project at Sandia National Laboratories is to provide a large-scale, massively parallel computing resource composed of commodity-based PC's that not only meets the level of compute performance required by Sandia's key applications, but that also meets the levels of usability and reliability of past traditional large-scale parallel machines. Cplant™ is a continuation of research into system software for massively parallel computing on distributed-memory message-passing machines. We have transitioned our scalable system software architecture developed for large-scale massively parallel processing machines to commodity clusters of PCs.

In this paper, we examine the performance and scalability of one of Sandia's key applications on Cplant™. We present performance and results from Salinas, a general-purpose, finite element structural dynamics code designed to be scalable on massively parallel processing machines.

The rest of this paper is organized as follows. The following Section provides an overview of the ASCI/Red machine. Section 3 describes the hardware and software environment of the flagship Cplant™ cluster. We present the details of the Salinas in Section 4, and present performance data in Section 5. The paper continues by discussing an important issue with scaling Salinas on Cplant™ in Section 6, and we summarize this paper in Section 7.

2 ASCI/Red

The Department of Energy's Accelerated Strategic Computing Initiative (ASCI) Option Red machine was installed at Sandia in the Spring of 1997. It is the culmination of more than ten years of research and development in massively parallel distributed-memory computing by both Intel and Sandia. The following describes the hardware and software components of ASCI/Red.

2.1 Hardware

ASCI/Red [14] is composed of more than nine thousand 333 MHz Pentium II Xeon processors connected by a network capable of delivering 400 MB/s unidirectional communication bandwidth. Each compute node contains two processors and 256 MB of main memory. The nodes are arranged in a 38x32x2 mesh topology. Each compute node has a network interface chip (NIC) that resides on the memory bus, allowing for low-latency access to all of physical memory on a node. The MPI half round trip latency for a zero-length message is 13 μ sec, while the asymptotic one-way MPI bandwidth is 311 MB/s.

2.2 Software

The compute nodes of ASCI/Red run a variant of a lightweight kernel, called Puma [15], that was designed and developed by Sandia and the University of New Mexico. The Puma kernel was originally developed on a 1024-processor nCUBE-2 and later ported to an 1800-node Intel Paragon. Intel and Sandia worked together to port Puma to the Intel x86 processor architecture for ASCI/Red, at which point it was productized and renamed Cougar by Intel. Cougar consumes approximately one percent of the total main memory on a compute node.

A key component of the design of Puma is a high-performance data movement layer called Portals. Portals were first implemented in SUNMOS to address the need for zero-copy message passing, where incoming messages are delivered directly into an application's address space without intermediate buffering by the operating system. Puma implemented the second generation of Portals (Portals 2.0), which extended the functionality of the original design and provided the basic building blocks for various high-level message-passing layers.

3 Computational Plant

The Computational Plant is a large-scale, massively parallel computing resource composed of commodity computing and networking components. The main goal of the project is to construct a commodity cluster capable of scaling to the order of ten thousand nodes to provide the compute cycles required by Sandia's critical applications. Because of this scalability requirement, CplantTM has been designed to address scalability in every aspect of the hardware and software architectures.

3.1 Hardware

Figure 1 illustrates the flagship CplantTM cluster at Sandia, called Antarctica. This cluster is composed of a large center section and four independent cluster heads. At any given time, the center section is attached to one of these heads and can be easily switched between any of the four. Currently the center section is composed of 1536 compute nodes. The three production cluster heads each have 256 compute nodes, and the development head contains 128 compute nodes. The production heads are distinguished by their external network connection to either Sandia's open network, restricted network, or classified network. This is very similar to the model used for the ASCI/Red supercomputer at Sandia, which has a center section that is able to be switched between unclassified and classified heads. This flexibility allows us to easily increase or decrease compute resources based on programmatic needs and schedules. The entire Antarctica cluster is composed of various compute node hardware purchased over the span of three years. Since the performance results for this paper were captured using the restricted network head in combination with the center section, we will only describe that set of hardware.

The compute nodes in the center section of Antarctica are composed of Compaq DS10L workstations that contain a 466 MHz Alpha EV67 processor, 256 MB of main

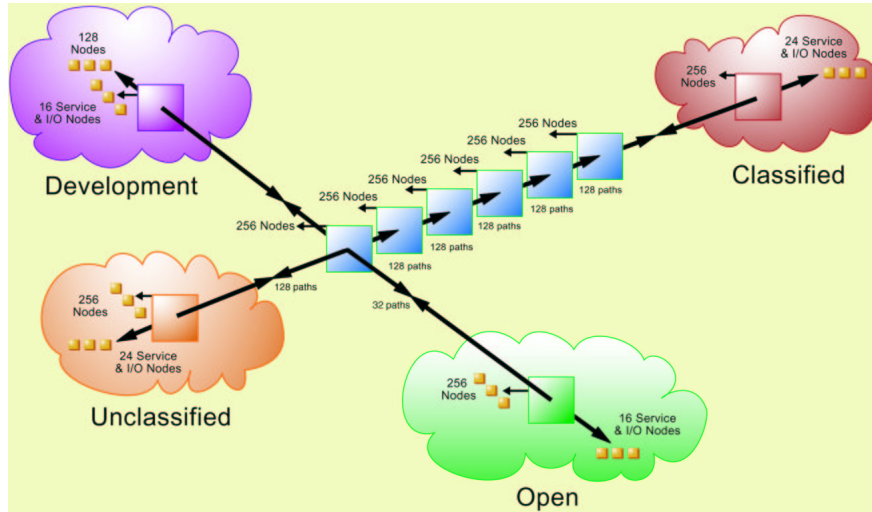


Figure 1: The Antarctica Cplant™ machine.

memory, and a Myrinet network interface card. There are a mixture of LANai version 7 and version 9 cards throughout the center section. The Myrinet network is a three-dimensional mesh that is torus in the X and Y directions but not in the Z direction. This topology is constructed using Myrinet Mesh-64 switches, which were a modification to their standard Clos-64 switches to provide an internal mesh in the switches. The mesh is constructed using 4 switches to build a single plane of 256 compute nodes in a 16x16 configuration, where each plane has 128 links in the Z- and Z+ directions. This structure allows us to easily add compute nodes in 256-node planes. The restricted network head cluster, named Ross, is a single plane of 256 compute nodes, each of which is a Compaq DS10L that contains a 617 MHz Alpha EV67 processor, 1 GB of main memory, and a Myrinet LANai-7 or LANai-9 network interface card. The MPI half round trip latency for a zero-length message is 65 μ sec, while the asymptotic one-way MPI bandwidth is 100 MB/s.

3.2 Software

The Cplant™ machines employ the partition model of resource provision [8] that was initially developed by Intel on their early parallel platforms. This model divides the machine into several different partitions that provide specialized functionality. The main partitions are service, compute, and I/O.

Nodes in the service partition run a standard Linux environment where users can log in and perform the usual UNIX commands, such as compiling codes, editing files, or sending email. The service partition is also where the users launch parallel applications into the compute partition, view status of running jobs, or debug compute node applications. The service partition is what many workstation clusters call the “front

end.” The service partition is composed of several different machines, and we employ a load balancing name server to place new logins on the least loaded machine.

The largest portion of the machine is the compute partition, which is dedicated to delivering processor cycles and interprocessor communications for parallel applications. Nodes in the compute partition are space-shared such that a group of nodes is dedicated to running only a single application. Our compute nodes are currently running a Linux 2.2.18 kernel and supporting software from a RedHat Linux 6.2 distribution. We have made some effort to keep the number system processes on the compute nodes small, in order to maximize the resources given to parallel application processes. User logins directly to compute nodes are prohibited.

The Cplant™ runtime system [5] is also modeled after the runtime system of ASCI/Red. This runtime system is dependent upon an underlying high-performance system area network, not only for supporting application message passing via a user-level library, such as MPI[12], but also for supporting compute node allocation, application launch, parallel I/O, and debugging tools. Cplant™ clusters use the Myrinet [2] gigabit network with the next generation implementation of Portals [3]. All communication between the runtime system components is implemented over Portals.

4 Salinas

Salinas offers static, direct implicit transient, eigenvalue, and sensitivity analysis capabilities. It also includes an extensive library of standard one-, two-, and three-dimensional elements, nodal and element loading, and multi-point constraints. Salinas solves systems of equations using an iterative, multilevel solver, which is specifically designed to exploit massively parallel machines. Salinas uses a linear solver that was selected based on the criteria of robustness, accuracy, scalability, and efficiency.

Salinas employs a multilevel domain decomposition method, Finite Element Tearing and Interconnect (FETI), which has been characterized as the most successful parallel solver for the linear systems applicable to structural mechanics. FETI is a mature solver, with some versions used in commercial finite element packages. For plates and shells, the singularity in the linear systems has been traced to the subdomain corners. To solve such linear systems, an additional coarse problem is automatically introduced that removes the corner point singularity. FETI is scalable in the sense that as the number of unknowns increases and the number of unknowns per processor remains constant, the time to solution does not increase. Further, FETI is accurate in the sense that the convergence rate does not deteriorate as iterates converge. Finally the computational bottleneck in FETI, a sparse direct subdomain solve, is amenable to high performance solution methods. An eigensolver was selected for Salinas based on robustness, accuracy, scalability, and efficiency.

Salinas is mostly written in C++ and uses 64-bit arithmetic. It combines a modern object-oriented FE software architecture, scalable computational algorithms, and existing high-performance numerical libraries. Extensive use of high-performance numerical building block algorithms such as the Basic Linear Algebra Subprograms (BLAS), the Linear Algebra Package (LAPACK), and the Message Passing Interface (MPI) has resulted in a software application which features not only scalability on thousands of

processors, but also a solid per-processor performance and the desired portability.

4.1 Distributed Architecture and Data Structures

The architecture and data structures of Salinas are based on the concept of domain decomposition. They rely on the availability of mesh partitioning software such as CHACO [9], TOP/DOMDEC [7], METIS [10], and JOSTLE [16]. The distributed data structures of Salinas are organized in two levels. The first-level supports all *local* computations as well as all interprocessor communication between neighboring sub-domains. The second level interfaces with those data structures of an iterative solver which support the *global* operations associated with the solution of coarse problems.

4.2 Analysis Capabilities

Salinas includes a full library of structural finite elements. It supports linear multiple point constraints (LMPCs) to provide modeling flexibility, geometrical nonlinearities to address large displacements and rotations, and limited structural nonlinearities to incorporate the effect of joints. It offers five different analysis capabilities: (a) static, (b) modal vibration, (c) implicit transient dynamics, (d) frequency response, and (e) sensitivity. The modal vibration capability is built around ARPACK's Arnoldi solver [11], and the transient dynamics capability is based on the implicit "generalized α " time-integrator [6]. All analysis capabilities interface with the same FETI-DP module for solving the systems of equations they generate.

5 Performance

Figure 2 shows the performance of Salinas on ASCI/Red and CplantTM out to 1000 nodes. We show the total time for Salinas as well as the time needed for the FETI portion of the code. A CplantTM compute nodes is roughly three times faster than a single processor on ASCI/Red. Despite the difference in network performance between ASCI/Red and CplantTM, Salinas runtime is roughly three times faster on CplantTM than ASCI/Red all the way to 1000 nodes.

6 Scaling Issue

The most important issue for scaling Salinas on CplantTM was dealing with MPI unexpected messages. Beyond several hundred processors, the MPI library would exhaust all of the space that had been allocated for buffering unexpected MPI messages and terminate the application. Increasing the amount of space for unexpected messages via a runtime switch helped to a certain point, but the problem consistently prevented Salinas from running up to 1000 nodes.

The initial MPI implementation for Portals on CplantTM set aside 1024 8 KB buffers, or 8 MB of memory, to buffer unexpected short messages which are sent eagerly. This number could be increased to a hard limit of 2048 by setting an environment variable.

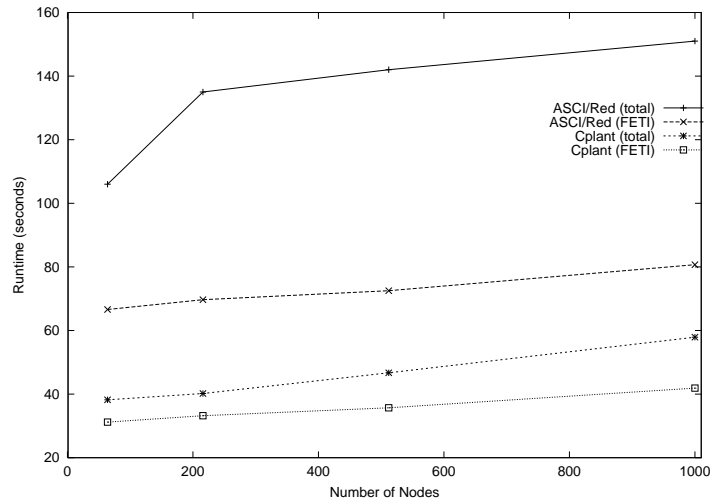


Figure 2: Salinas performance on ASCI/Red and Cplant™.

However, in the initial implementation, a single 8 KB buffer could be consumed by a zero-length unexpected message. Therefore, the unexpected message space could be “consumed” by 2048 messages of any size. The semantics of Portals did not allow efficient use of the memory that had been dedicated to unexpected buffers.

The source of the unexpected messages turned out to be the `MPI_Gather()` routine, which in the MPICH 1.2.0 implementation of MPI is implemented as by having all ranks send their contribution of the gather directly to the root process. Since these gathers were short eager messages, the non-root ranks would make their contribution and immediately continue. Successive `MPI_Gather()` calls could swamp the root process and exhaust the unexpected message buffers when run on several hundred nodes. This naive non-scalable N to 1 implementation of `MPI_Gather()` is clearly not optimal.

The initial workaround to scaling Salinas up to 1000 nodes was to add an `MPI_Barrier()` call after each `MPI_Gather()` to insure that the only a single gather operation could be in progress at any given time. This workaround was used in gathering the performance data in this paper.

We have since made a small semantic change to Portals and enhanced our MPI implementation for Cplant™ to make more effective use of MPI unexpected message buffer space. This new strategy allows for the number of unexpected messages to be dependent on space rather than count, so there is less restriction on the number of unexpected messages that can arrive. For a more complete description of the MPI implementation on Cplant™ and these changes for unexpected messages, see [4]. We plan to gather performance data for this new MPI implementation to try to measure the impact of the extra barrier operations in the data presented here.

7 Summary

In this paper, we have described the large-scale Cplant™ cluster and described Salinas, one of our important applications. We have compared performance results to a traditional supercomputer, the ASCI/Red machine. We have detailed the hardware and software environments of both platforms, including the unique ability of the Antarctica Cplant™ to switch a large portion of compute nodes between four independent cluster heads. We have presented performance results for Salinas on 1000 nodes on both machines that indicate that Salinas scales as well as the traditional supercomputer. We discussed some limitations of the software environment that initially prevented us from running Salinas on 1000 nodes and discussed our workaround and fixes.

References

- [1] F. Berman. Viewpoint: From teragrid to knowledge grid. *Communications of the ACM*, 44(11):27–28, 2001.
- [2] N. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet-a gigabit-per-second local-area network. *IEEE Micro*, 15(1):29–36, February 1995.
- [3] R. Brightwell, W. Lawry, A. B. Maccabe, and R. Riesen. Portals 3.0: Protocol Building Blocks for Low Overhead Communication. In *Proceedings of the 2002 Workshop on Communication Architecture for Clusters*, April 2002.
- [4] R. Brightwell, A. B. Maccabe, and R. Riesen. Design and Implementation of MPI on Portals 3.0. In *Proceedings of the Ninth EuroPVM/MPI Conference*, September 2002. To appear.
- [5] R. B. Brightwell and L. A. Fisk. Scalable Parallel Application Launch on Cplant™. In *Proceedings of the SC2001 Conference on High Performance Networking and Computing*, November 2001.
- [6] J. Chung and G. M. Hulbert. A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized- α Method. *Journal of Applied Mechanics*, 30(371), 1993.
- [7] C. Farhat, S. Lanteri, and H. D. Simon. TOP/DOMDEC a software tool for mesh partitioning and parallel processing. *Comput. Syst. Eng.*, 6:13–26, 1995.
- [8] D. S. Greenberg, R. B. Brightwell, L. A. Fisk, A. B. Maccabe, and R. E. Riesen. A System Software Architecture for High-End Computing. In *Proceedings of SC'97*, 1997.
- [9] B. Hendrickson and R. Leland. The Chaco User's Guide: Version 2.0. Technical Report SAND94-2692, Sandia National Laboratories, 1994.
- [10] G. Karypis and V. Kumar. Parallel Multilevel k-way Partition Scheme for Irregular Graphs. *SIAM Review*, 41(2):278–300, 1999.

- [11] R. Lehoucq, D. C. Sorensen, and C. Yang. *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods*. SIAM, 1998.
- [12] Message Passing Interface Forum. MPI: A Message-Passing Interface standard. *The International Journal of Supercomputer Applications and High Performance Computing*, 8, 1994.
- [13] Pacific Northwest National Lab. *PNNL Orders \$24.5M Supercomputer from Hewlett-Packard*. <http://www.pnl.gov/news/2002/computer.htm/>.
- [14] Sandia National Laboratories. *ASCI Red*, 1996. http://www.sandia.gov/ASCI/TFLOP/Home_Page.html.
- [15] P. L. Shuler, C. Jong, R. E. Riesen, D. van Dresser, A. B. Maccabe, L. A. Fisk, and T. M. Stallcup. The Puma operating system for massively parallel computers. In *Proceedings of the 1995 Intel Supercomputer User's Group Conference*. Intel Supercomputer User's Group, 1995.
- [16] C. Walshaw and M. Cross. Parallel Optimization Algorithms for Multilevel Mesh Partitioning. *Parallel Computing*, 26:1635–1660, 2000.